Geo-Fencing for Unmanned Aerial Vehicle

PL Pratyusha
Department of Avionics
Institute of Science and Technology, JNTUK
Kakinada, India
plpratyusha1@gmail.com

VPS Naidu
Multi-sensor Data Fusion (MSDF) Lab,
CSIR-NAL (National Aerospace Laboratories),
Bangalore, India
vpsnaidu@gmail.com

ABSTRACT
Geo-fencing is an application which plays a major role in security for Unmanned Aerial Vehicles (UAVs) and Micro Aerial Vehicles (MAVs). This application can be used in Ground Control Station (GCS), which helps in increasing the level of security, or it can be used on board UAV to increase the level of automation. The current work includes two types of geo-fencing (polygonal and circular) techniques for GCS. The geo-fencing algorithm has been implemented and tested using Unmanned Aerial System (UAS) simulator which consist of UAV Flight Simulator (UAVFS), Communication link (CL) and GCS. A Graphical User Interface (GUI) has been designed which helps in user interaction and the simulated results are satisfactory. The geo-fencing algorithm is developed in Visual C#.Net environment.

General Terms
Algorithm, Security, MAV, Software

Keywords
Unmanned Aerial Vehicle (UAV), Ground Control Station (GCS), Geo-fencing, GUI, C#.Net, UAV simulator

1. INTRODUCTION
Geo-Fencing is a boundary or region of interest in the geographical region. Geo-fencing is used for many applications and it provides many benefits to users. One of the major applications for Geo-fencing is security, when anyone enters or leaves a particular area, an alert or text messages has been sent to the user [1]. In military applications this can be used, when the enemy vehicles enter into our boundary it gives an alarm sound to alert the officers and takes necessary actions.

Geo-fencing is necessary option for GCS as well as for onboard UAVs which helps in increasing the level of autonomy. When UAV goes out of range or if it crosses a particular boundary defined by the user, geo-fencing is used to give an alert to the pilot/operator. For example, an operator who doesn't know the limit of caution region can fly beyond it, in such cases geo-fencing algorithm will give the sound alerts so that user can operate it safely. If UAV is flying in auto mode, with the help of algorithm UAV will not go beyond this caution region. Geo-fencing is also used for restricted areas; if anyone enters into those regions then it gives an alarm.

In critical operational scenario, a boundary is drawn by the user to confine the UAV movement. If the UAV crosses the boundary then an alarm sound with a message box will be displayed. The boundary can be polygon, circle etc. In polygon type geo fencing, the user will select different points and a polygon is drawn between those points and checks whether the UAV is within the boundary or not. In circle type, user has to give the radius. Once the radius is entered by the user, a circle with that radius is drawn at user's point of interest. The second section explains about the mathematical logic behind the geo-fencing algorithm. Third section gives a detail explanation about experimental setup. Finally, Fourth section discuss about the simulation results.

2. Geo-fencing Algorithm
This paper includes two types of geo-fencing algorithms namely polygonal geo-fencing and circular geo-fencing. C#.Net based GUI is developed for this algorithm which requires a map [2] and provides a selection for type of boundary. Some additional functions are also included like save the data in XML format and load the saved data, selecting the type of map etc.,

2.1 Polygonal Geo-fencing
A Polygon is drawn between user selected points. Geo-fencing algorithm uses those selected points defines the boundary and determine whether the UAV position is inside boundary or not. Polygon may be of any shape, for complex type of polygons it is difficult to find whether the point is inside or outside the boundary. The solution for this type of polygon is, make a horizontal threshold line which is passing through the UAV position [3]. Each side of polygon is taken as y coordinate of UAV position and each side is represented as a node. If there are an odd number of nodes in each side of UAV position then it is within the boundary, if there is an even number of nodes in each side then it is out of boundary. If the UAV is exactly in the polygon edge then it may be within or out of the boundary [3]. The C# code for determining whether UAV position is within or outside the polygon is:

```
Public bool FindUAVPosition(doubleX,
double Y)
{
int no_of_polySides =this.Count( )-1;
int j = no_of_polySides -1;
bool UAVStatus=false;

//Position of UAV is  (X,Y)
for(int i =0; i < no_of_polySides; i++)
{
// checks Y coordinate in range of
bndryPts[i], bndryPts[j]
if(bndryPts[i].Y < Y &&bndryPts[j].Y >= Y
||bndryPts[j].Y < Y &&bndryPts[i].Y >= Y)
{
// checks X coordinate below the line
which is drawn between points i and j
if(bndryPts[i].X +(Y -bndryPts[i].Y)/(
bndryPts[j].Y -bndryPts[i].Y)*(
bndryPts[j].X -bndryPts[i].X)< X)
```

```
{
UAVStatus =! UAVStatus;
}
}
          j = i;
}
return UAVStatus;
}
```

In the above code, initially it will check the Y coordinate of UAV position within the range of (bndryPts[i], bndryPts[j]), if it is true it will check whether the X coordinate of UAV position is below the line, which is connecting from point i to point j, or not. If these two conditions are true for odd number of times then 'FindUAVPosition( )' function will return true otherwise it returns false. If it is true then UAV is inside otherwise it is outside.

Let us consider an example as shown in the Fig 1. From this figure a threshold (brown dotted line) is taken which passes through UAV position (red dot), each side of polygon which touches the threshold taken as a node. If there is odd number of nodes on either side of UAV position, then UAV is within the boundary [2]. If the number of nodes is even, then UAV is in outside of the boundary. In this example three nodes are on the left side of UAV, hence UAV is inside the boundary. The example is for static point (i.e. assuming that UAV is at constant point) and algorithm is working correctly. Now the problem is, UAV will move continuously, the latitude and longitude values has to update and it has to verify whether the UAV is inside or not. By using timer, receiving data event was called continuously. Then the UAV position data is passed continuously to another form [4]. The C# code for sending the updated position of UAV from Form1 to Form 2 is shown below. In Form 1 code, whenever user clicks on the 'polygon' item from context menu, a timer will get started and updates the UAV position for every 100ms. In Form 2, data_ received( ) function will get the UAV position continuously as shown in the Fig 2.

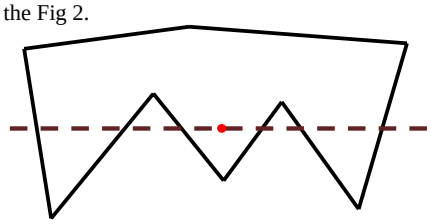

Fig 1: Example of polygonal Geo-fencing [3]

Form1:

```
private void
rectangleToolStripMenuItem_Click(objectse
nder,EventArgs e)
{
childfrm_fencing=newGeo_Fencing();
childfrm_fencing.Show();//opens    another
form
timer1.Start();
}
Private void
timer1_Tick(objectsender,EventArgs e)
{
ReceivingData+=newDataHandler(childfrm_fe
ncing.data_received);
storeddata();
}
Private void storeddata()
{
PointLatLng data;
var lat2 =Interface.UAVPose.lat;
var lng2 =Interface.UAVPose.lon;
data=newPointLatLng(lat2, lng2);
DataEventArgs e =newDataEventArgs();
e.LatLng_frm2 = data;
//Lat,Lng values are stored in event args
sendData_frm2(newobject(),e);
}
Private void
sendData_frm2(objectp,DataEventArgs e)
{
if(ReceivingData!=null)
{
ReceivingData(newobject(), e);
}
}
```

Form2:

```
Public void
data_received(objectsender,DataEventArgs
e)
{
textBox1.Text = e.LatLng_frm2.ToString();
gMapfence1.Position = e.LatLng_frm2;
}
#
```

Fig 2: Example of Polygon geo-fencing at HAL Airport

2.2 Circular Geo-fencing

If circular fencing radio button was clicked, an input text box will appear as shown in the Fig 3, in that user has to give the value of radius in meters.

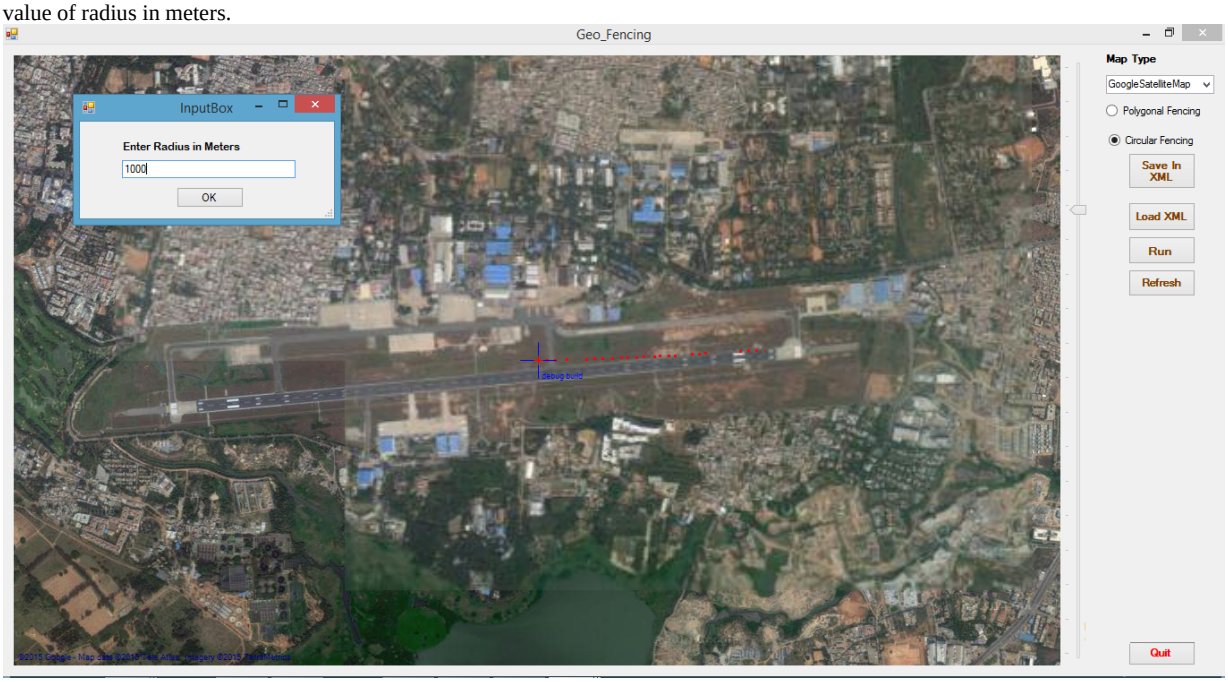

Fig 3: Message box taking radius as input

With that radius, a circle will be drawn at the user point of interest in the map. This circle will be taken as boundary to the UAV Position and if it crosses the boundary then a voice alert as well as message box will appear.

Circular geo-fencing is easier compared to polygonal geo-fencing in terms of implementation. In circular geo-fencing, distance between centre of the sphere and current UAV position has to be calculated, if the distance is greater than radius then the UAV crosses the boundary. If the distance is less, then UAV is inside the boundary. In order to calculate the great circle distance between two points on sphere "Haversine Formula" is used [5]. In earlier trigonometric tables $ver\sin(\theta)$ is a trigonometric function, which equals to

$$versin(\theta) = 1 - \cos(\theta) = 2\sin^2\left(\frac{\theta}{2}\right) \quad (1)$$

Whereas $haversin(\theta)$ is half the $versin(\theta)$ i.e.,

$$haversin(\theta) = \frac{1-\cos(\theta)}{2} = \sin^2\left(\frac{\theta}{2}\right) \quad (2)$$

Haversine formula for calculating distance between two points on sphere is given by the equation

$$haversin\left(\frac{d}{r}\right) = haversin(d\phi) + \cos(\phi_1)\cos(\phi_2)haversin(d\lambda) \quad (3)$$

Where, ϕ_1, ϕ_2 are latitudes of point 1 and point 2

λ_1, λ_2 are longitudes of point 1 and point 2

d is the distance between two points (along the sphere) in km

r is the radius of earth in km = 6371 km

$d\phi = \phi_2 - \phi_1$

$d\lambda = \lambda_2 - \lambda_1$

d can be obtained with the help of sine inverse ($\arcsin$) function

$$d = r * haversin^{-1}(h) \quad (4)$$

Where

$$h = haversin\left(\frac{d}{r}\right) = \sin^2\left(\frac{d}{2r}\right) \quad (5)$$

By solving the above equation (Eq.5)

$$d = 2r\sin^{-1}\left(\sqrt{h}\right) \quad (6)$$

$$d = 2r\arcsin\left(\sqrt{\sin^2\left(\frac{d\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{d\lambda}{2}\right)}\right) \quad (7)$$

Eq.7 is the final equation to calculate distance between two points on a sphere in kilometres in which the latitudes and longitudes has to be converted to radians (i.e., multiplying the latitude/longitude value with pi/180). If this result is less than radius of the circle then UAV is inside the boundary else it is out of boundary. C# code implements the above formula for calculating the distance is shown below:

```
public double Distance(PointLatLng pos1,
PointLatLng pos2)
      {
int Radius = 6371;   // radius of earth
in km
double dLat = this.toRad(pos2.Lat-
pos1.Lat);
double dLon = this.toRad(pos2.Lng-
pos1.Lng);
double inn_brckt = Math.Sin(dLat / 2) *
Math.Sin(dLat / 2) +
Math.Cos(this.toRad(pos1.Lat)) *
Math.Cos(this.toRad(pos2.Lat)) *
Math.Sin(dLon / 2) * Math.Sin(dLon / 2);
double f = 2 * Math.Asin(Math.Min(1,
Math.Sqrt(inn_brckt)));
double d = Radius * f;
return d;
}
private double toRad(double deg_val)
{
   return (Math.PI / 180) * deg_val;
}
```

After calculating the distance, if the distance is greater than radius then it will alert the operators by giving warning sound. C# code for checking the UAV whether it is inside or not is:

```
if(radbtn_circle.Checked == true &&
radbtn_poly.Checked != true)
{
PointLatLng LL_circle =
gMapfence1.Position;
double dist_result =
Distance(circle_centre, LL_circle);
    if(dist_result > (radius/1000))
  {
Warningsound.Play();
MessageBox.Show("UAV Crossed boundary",
"Error", MessageBoxButtons.OK);
  }
}
```

3. Experimental Setup

A laboratory based UAS simulator setup has been developed to test the geo-fencing algorithm at MSDF lab, CSIR-NAL. The UAS has been realized with two PC's as shown in Fig 4.

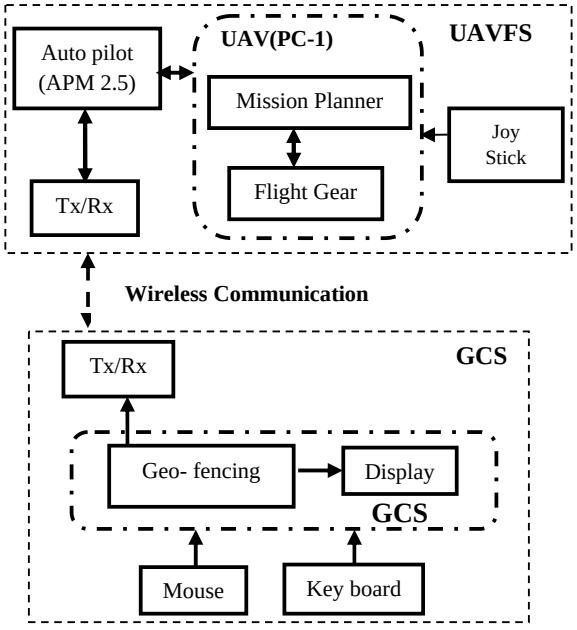

Fig 4: Block diagram of UAS setup

UAS consists of the following three components:

- **UAV Flight Simulator (UAVFS)**
  It is difficult to fly an aerial vehicle for every time to check the GCS functions, hence a UAVFS is

required. UAV Flight simulator is a PC -1 (which consists of Flight gear and Mission Planner).

- **Wireless Communication link**
  Wireless Communication Link plays a crucial role in UAS. Because, transferring of data from UAV to GCS is done through this link. This link should be bi-directional. UAV can send the data regarding its position, attitude, speed etc., at the same time it has to receive the commands from the GCS. Similarly, GCS should be capable of receiving and sending commands to UAV. In UAVFS, through flight gear the data will be transmitted to the APM board which transmits the data to GCS through X-Bee Pro modules.

- **Ground Control station (GCS) section**
  GCS is a PC-2 which is having Geo-fencing algorithm.

3.1 UAVFS

PC-1 is act as UAVFS. PC-1 has following specification

- Processor: Intel(R) Core(TM) i7 CPU 870 @ 2.93GHz
- RAM: 4 GB
- System type:64 - bit Windows 7 OS
- Graphics card : NVIDIA GeForce 8400GS

For making PC-1 as UAVFS it should have following software and hardware.

1) Software Required in PC-1

a) Mission Planner-1.3.10 (needs .NET framework version v4.0) [6]

b) Flight Gear(complied with generated photo scenery)

c) APM board drivers [7]

2) Hardware Required in PC-1

a) APM 2.5 (Ardu Pilot Mega board)

b) XBee (able to connect with APM 2.5)

Steps for configuring UAVFS

1. Launch the Mission Planner Software by double clicking on the Mission Planner icon placed on the desktop (PC-1).

2. Once the Software is launched the dialog box opens in which we need to select the port to which the Autopilot board is connected as shown in Fig 5(a) and then enter the baud rate as 115200.Click on Connect to connect to APM 2.5 board.

Fig 5(a): Initializing the Mission Planner

3. Then Mavlink will be established as soon as the connect button is clicked. Once the connection is established, on the left hand side of the window the status will be changed from DISARMED to ARMED as shown in Fig 5(b), which indicates that the device is connected.

Fig 5(b): MAVLINK Connection established

4. Then Click on Configuration > Basic Tuning, check for the P to T (0.100) and Rudder Mix's (0.200) Values shown in Fig 5(c). In case they are different, change the values to specified values [6, 7].

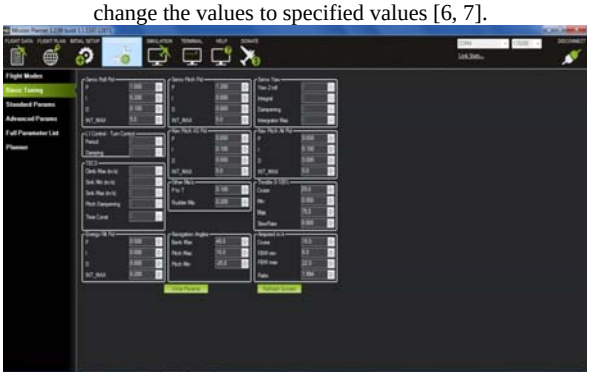

Fig 5(c): Tuning the UAV in configuration tab

5. Click on Flight Modes and check for Mode 1 and Mode 2 to be Auto and Mode 3 and Mode 4 to be STABILIZED and Mode 5 to be Manual as shown in Fig 5(d).

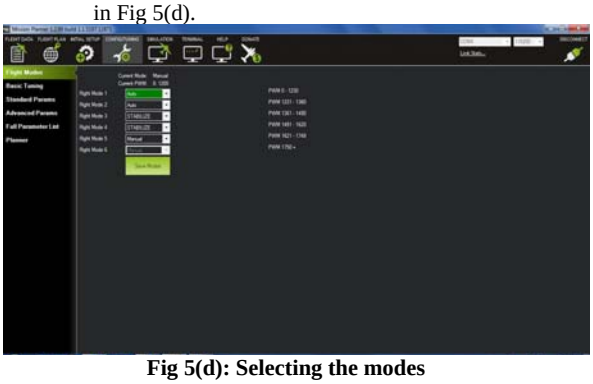

Fig 5(d): Selecting the modes

6. Then go to Flight Plan, select the waypoint (Wp). Set default Alt as 50 and click on absolute Alt. For the starting point, select it as take off, it should be

home and decrease the Alt to 10 and for last waypoint select it as land and its Alt should be 0.Then click on Write Wp's which will write the Waypoints on the board [8] as shown in Fig 5(e).

Fig 5(e): Writing waypoints onto the log

7. Then click on Simulation, on the Simulation window select Flight Gear radio button as shown in the Fig 5(f) and then click Start FG Plane. It will open a "Flight Gear" window which indicates the movement of UAV as shown in Fig 5(f).

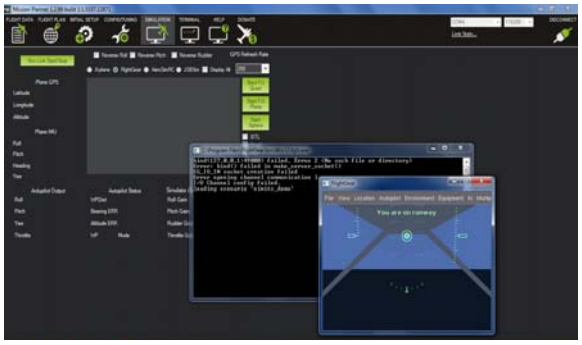

Fig 5(f): Flight gear simulation

8. Then click on Sim Link Start/Stop which will start the simulation as shown in Fig 5(g). Once simulation started user should get following screen or else check the configuration of Flight Gear.

Fig 5(g): Start SimLink

3.2 GCS

PC-2 is act as GCS-Geo fencing PC. PC-2 has following specification

- Processor: Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz
- RAM: 4 GB
- System type:64 - bit Windows 8.1 Pro

Steps to Configure

1. Launch the GCS Software.
2. Once the Software is launched, at the top right corner there are two combo boxes in which select the port to which the X-Bee Pro module is connected and then select the baud rate as 57600.Click on Connect button to open the port and receive the data.
3. Mavlink will be established as soon as the connect button is clicked. Once the mavlink is established, the Connect button will be changed to Connected as shown in Fig 6.

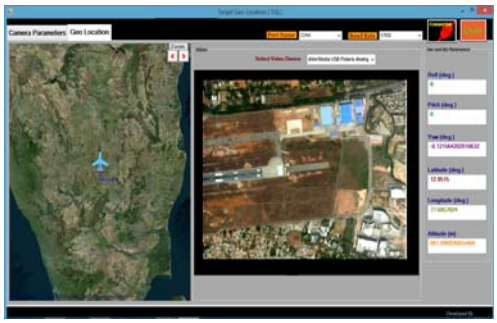

Fig 6: GCS Software

4. Continuous data is received and updated which is shown in the right most side of the window as shown in Fig 6. According to that data, the position of the aircraft in the map is also updated.
5. Right click on the map then a context menu strip will open as shown in the Fig 7, in that select the Geo-fencing > polygon option which opens another window as shown in Fig 8 and performs required operations.

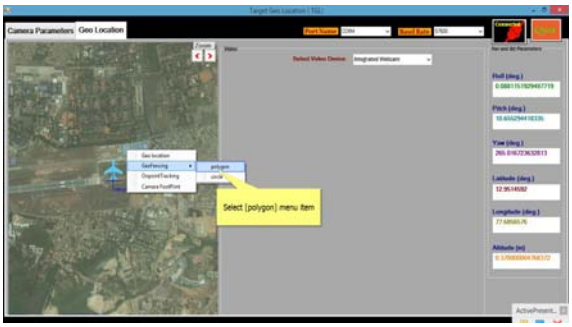

Fig 7: Starting the Geo-fencing window

Fig 8: Geo-fencing Window

The realized model at MSDF Lab, CSIR-NAL with the above three sections is shown in the Fig 9.

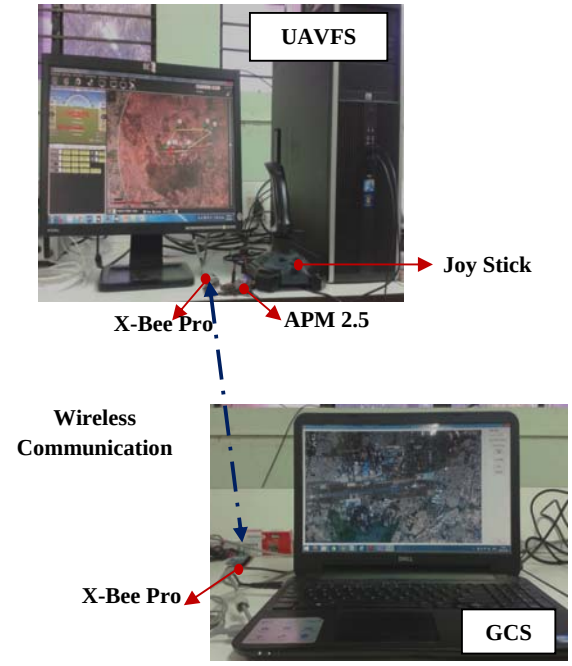

Fig 9: Realized model of UAS

4. Results and Discussions

Initially, start the flight simulator as explained in the previous section, once simulation gets started, the flight can be controlled either by manually through joystick or by auto mode. The data has been transmitted from UAVFS to GCS through APM2.5 & XBee Pro RF module. When user selects the geo-fencing option in GCS, a new window will be open as explained in section 3.2. In that Geo-fencing form, select the type of the map from the right hand side combo box and then select the type of fencing as polygonal as shown in Fig 10.

Fig 10: Polygonal Fencing

The red dots in Fig 11 are UAV's updated positions; if the "Run" button is clicked then it will check whether the UAV position is within or without the polygonal boundary. If UAV crosses the polygon then it will give an error message box and a warning sound will play as shown in Fig 11 and the pseudo code of polygonal fencing is given in the section 2.1.

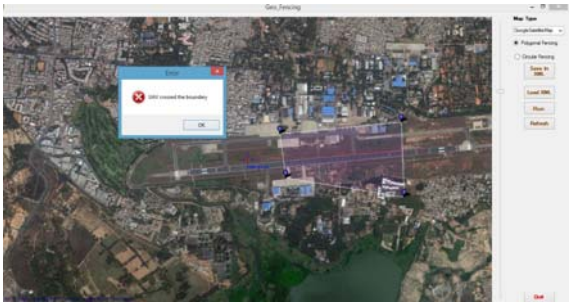

Fig 11: The simulated data (UAV position) crosses the boundary of polygon

In circular type of fencing, user will give the radius of circle in meters as shown in the Fig 12

Fig 12: Circular Fencing

With that radius a circle will be drawn, if the UAV crosses the circular boundary then an alert sound with a message box will be displayed as shown in the Fig 13 and the pseudo code of circular fencing is given in the section 2.2.

By clicking on the "Save in XML" button, user can save the boundary either polygon or circular in a XML file for future use. Load these saved data through "Load XML" button. "Refresh" button will refresh the map and removes the planned boundaries.

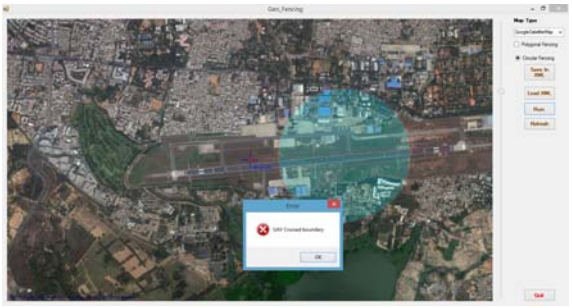

Fig 13: The simulated data (UAV position) crosses the boundary of circle

5. Conclusion

As part of GCS, a GUI has been developed for Geo-fencing in C# .Net environment. Using this algorithm, a virtual boundary can be drawn using polygon or circular methods on the map at GCS to confine the UAV flying. While flying, UAV sending its position through telemetry link to GCS and Geo-fencing algorithm checks whether UAV crosses the boundary or not, and if UAV crosses the boundary, an audio as well as pop-up message will alert the operator at the GCS. It is observed that circular fencing is easy compared to polygonal fencing in terms of implementation. Provision has been added to GUI to save the planned geo-fencing scenario as xml file for future use. There is a provision to load previously planned geo-fencing scenario.

6. ACKNOWLEDGMENTS

We would like to thank Miss. Indhu B for helping us while dealing with hardware.

7. REFERENCES

[1] Andrew, "How to make Geo-fencing work for your small business", http://www.biznessapps.com/blog /2014/03/18/how-to-make-geofencing-work-for-your-small-business/, 2014, accessed on 20th March 2015.

[2] "Using GMap.Net- Great Maps for Windows Forms & Presentation", http://geekswithblogs.net/saifkhan/archive/2011/08/03/using-gmap.net-ndash-great-maps-for-windows-forms-amp-presentation.aspx, 2011, accessed on 14th November 2014.

[3] Darel Rex Finley, "Point-in-Polygon Algorithm – Determining whether a Point is inside a complex polygon", http://alienryderflex.com/polygon/, 2007, accessed on 18th March 2015.

[4] MitjaBonca, "Passing continuous data between multiple displayed forms", https://social.msdn.microsoft.com/Forums/vstudio/en-US/d09a814b-d943-4d29-a185-3748da14f371/passing-continuous-data-between-multiple-displayed-forms?forum=csharpgeneral, 2011, accessed on 18th March 2015.

[5] "Haversine Formula", http://en.wikipedia.org/wiki/Haversine_formula, accessed on 18th March 2015.

[6] "Mandatory APM 2.5 Hardware Configuration", http://copter.ardupilot.com/wiki/initial-setup/ configuring- hardware/, accessed on 24th November 2014.

[7] "Mission Planner", http://planner.ardupilot.com/, accessed on 24th November 2014.

[8] "Planning a Mission with Waypoints and Events", http://planner.ardupilot.com/wiki/ common -planning-a-mission-with-waypoints-and-events/, accessed on 24th November 2014.